California State University
Northridge

Department of Computer Science

THESIS/PROJECT PROPOSAL

TITLE: Automated Software Test Tool

Eugean Hacopians

File No: 932-1022

Date: June, 96

Approved: _____ _____

Committee Chair: Dr.Shan Barkataki

1 OBJECTIVE

The objective of the Graduate project described below is to introduce the development of a software tool which will be used to automate the testing process of a JPI,-specific set of software programs.

2 INTRODUCTION

The Multi-mission Ground System Office (MGSO), which is part of the Jet Propulsion Laboratory (JPL) organization, produces a multiple set of core software programs to assist in the generation of flight sequences that are uplinked to spacecraft through the Deep Space Network (DSN). MGSO develops these software programs by collecting all the common requirements from different JPL/NASA projects. Upon delivery of the MGSO core software, each project modifies the program and tailors it to their specific needs and requirements by manipulating the necessary files.

These programs are inter-linked together. For instance, the output of one software program is an input to another, in addition to passing initialization files such as the command database and flight rules. A Sequence Integration Engineer (SIE) may generate the initial input file by using one of these software tools.

One can look at these programs as an "operating system" of the spacecraft, but with some differences. Consider the following: when a UNIX command directive such as "1s" is entered at the command line, the result is a list of the current working directory. However, the steps involved in executing this command directive occur within the operating system and are transparent to the user. The "ls" command, after some translations, is converted into binary, loaded into CPU memory, and then executed. A similar process occurs on the spacecraft which carries the computer system (spacecraft brain) onboard. Due to size and weight limitations, there is a limited storage (hard disk) on board the spacecraft computer system. Therefore, only a portion of the operating system is installed onboard, while the remaining portion remains in the ground system. In other words, the command translations and binary conversions

remain in the ground system, and then command bits are uplinked to the spacecraft for command processing ant] execution.

These programs are large in size and complexity. Many files could be manipulated during the process of adaptation and, therefore, each software program must be tested at the unit level. Also, other initialization files are created during the adaptation phase either manually, by software or combination of both. These files must also be tested for completeness and correctness.

The process of testing these files is painstaking work. Due to the amount of data contained in each file, it is very difficult to test all possibilities. To simplify this process, a software utility tool, Automated Software Test Tool (ASTT), will be developed to test these programs and files in a more efficient method.

3 TECHNICAL APPROACH

The Automated Software Test Tool (ASTT) which will b c developed using Object-Oricntccl Design (OOD) and implemented i n Object-Oriented Programming (001'), is divided into two major parts. The first part will read a command database file containing a description of all valid spacecraft commands and create test case scenario files. The second part will run each test case scenario file through the software chain to verify the validity and correctness of every program and initialization file. If at any time, a program in the chain fails to produce a correct output file or fails altogether, the program will be terminated and the chain will be concluded. A report will be generated to indicate the success or failure of each test case.

The ASTT will produce approximately 1600 test cases, one per command. Each file will contain one command with all the permutation of parameter values. For instance, if a command has three parameters with the first parameter having four possible values, the second parameter having three possible values, and the third parameter having two possible values, then the test file would contain 4X3X2 = 24 instances of that one command. Command parameters range from zero to thirty, with an average of four

parameters per command. Each parameter has about two to twenty values, with an average of five values per parameter. To process a test of this magnitude, it would take approximately seven days of non-stop processing on a 100 Million of Instruction Per Second (MIPS) Hewlett Packard 735 workstation. To shorten the test duration, ASTT will have the capability to direct test execution tasks to multiple workstations based on users request. ASTT will have the ability to access 3 Hewlett Packard 735 workstations and 15 Hewlett Packard 725 workstations (that run at 50 MIPS). The user will interact with the ASTT via a Graphical User Interface (GUI).

This project will be accomplished in three incremental phases. The software for generating the test case scenario files will be developed in the first phase, along with the command database interface, and processing of each command. The second phase will include developing the supporting scripts which will be used to run the test files through the software chain for verification and validation. Also the preliminary part of the GUI will be developed in the second phase. Enhancing the capabilities of ASTT will be done in the third phase, such as distributing the processes over multiple workstations, enhancing the supporting scripts and GUI to support the process distribution. Each delivery phase will include a specified period of time allocated for testing the delivered products and their capabilities.

Work breakdown structure is as follows:

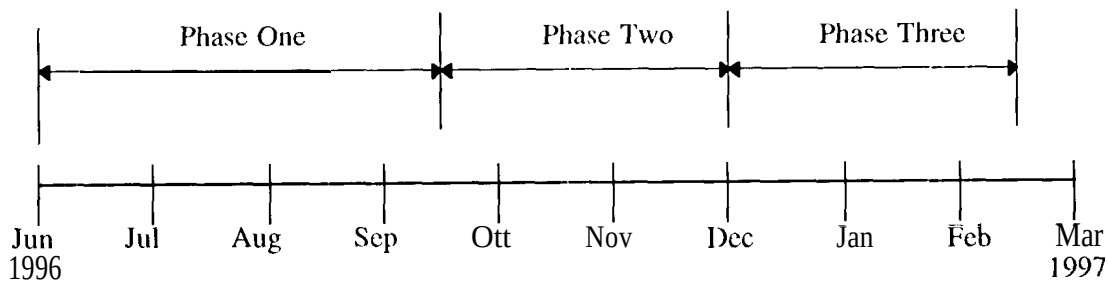
. Phase One
    1. Develop command database interface.
    2. Process commands.
    3. Create “test case scenario” files (which is the initial input file).
    4. Verify and validate phase one capabilities.

. Phase Two
    1. Develop scripts in order to run the test files through the software chain and detect any failure.
    2. Develop the preliminary part of the GUI.
    3. Verify and validate phase two capabilities.

- Phase Three
    1. Distribute the processes over multiple workstations.
    2. Enhance the supporting scripts and GUI to support the process distribution.
    3. Verify and validate phase three capabilities.

4 SCHEDULE

The elapsed time for this project will be June 1996 through February 1997. The chart below depicts the milestones.



Note: The phase and steps are defined in Section 3 of this document.

5 CRITERIA FOR SUCCESS

The minimum success criteria for validating and evaluating the ASTT are as follows:

1. Generation of test case scenario files using the command database interface.
2. Development of a major script file used to run a test case through the chain of software programs on one workstation.
3. Distinguishing between successful or unsuccessful "failed" test cases.
4. Availability of a user-friendly GUI that utilizes and monitors the success criteria 1, 2, and 3.

The desired success criteria for validating and evaluating the ASTT are as follows:

5. Distribution of processes onto multiple workstations.

6. Successful production and implementation of the ASTT to support users with effective and automated testing capabilities.

6 Reference

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. "Object-Oriented Modeling and Design." Prentice-Hall, New Jersey, 1991

[2] D. Norman. "The Design of Everyday Things." Doubleday, New York, 1989.

[3] B. Shneiderman. "Designing the User Interface." Addison-Wesley, New York, 1992. Second Eddition.

[4] R. Pressman. "Software Engineering: A Practitioner's Approach. " McGraw-Hill, New York, 1992. Third Edition.